IN THE UNITED STATES PATENT & TRADEMARK OFFICE

TITLE

~~FAULT TOLERANT STORAGE SYSTEM AND METHOD~~

INVENTORS

ALEXANDER TORMASOV
MIKHAIL KHASSINE
SERGUEI BELOUSSOV
STANISLAV PROTASSOV

Bibliography

Hamming, Richard W. 1986. *Coding and Information Theory*, 2nd Ed., Prentice-Hall Inc.

Patterson, David A., Garth A. Gibson and Randy H. Katz. *A Case for Redundant Arrays of Inexpensive Disks (RAID)* by David A. Patterson, Computer Science Division Department of Electrical Engineering and Computer Sciences, 571 Evans Hall University of California Berkeley, CA 94720 CSD-87-391.

Pfister, Gregory F. 1998. *In search of clusters*. Second edition. Prentice Hall, ISBN 0-13-899709-8.

Roman, Steven 1996. *Introduction to Coding and Information Theory (Undergraduate Texts in Mathematics)* Steven Romann, P. R. Halmos (Editor), Springer Verlag ISBN: 0387947043.

U.S. Patent Document 6,173,377 January 9, 2001 Yanai, et al. 711/162.

U.S. Patent Document 6,157,991 December 5, 2000 Arnon 711/161.

U.S. Patent Document 5,537,533 July 16, 1996 Staheli, et al. 714/5.

BACKGROUND OF THE INVENTION

This application claims the benefit of U.S. Provisional Application No. 60/269,452, titled "*A Method of Storing and Retrieving Information With Controllable Redundancy for Fault Tolerance Distributed Data Storage*" filed on February 16, 2001 for priority under 35 U.S.C. §119(e), is related thereto, is commonly assigned therewith, and incorporates herein by reference in its entirety the subject matter thereof.

Technical Field

This invention relates to the organization of a distributed data storage system, more particularly, the present invention relates to the storage and retrieval of information with controllable redundancy for fault tolerant distributed data storage.

Background

With the growth of the use of the Internet, the need for data storage systems with the capability to manage huge amounts of information has grown dramatically. Such data storage or information management systems must provide reliable service to millions of computer users simultaneously.

In prior art data storage networks, a large amount of data is broken into smaller pieces and transmitted using a store and forward mechanism.

Anyone deploying a data storage or information management system must deal with insufficient communication channel bandwidth and the inability of computer hardware components to handle the data storage load.

One prior art approach to solving the problems of insufficient bandwidth and the inability of computer hardware to store sufficient amounts of data has been to build a distributed network data storage system (Pfister 1998). In a typical distributed network data storage system, data is stored on a network of computers which consists of a mesh of data transmission links, switching nodes, and end nodes. The data pieces are transmitted on a series of links which connect the source of the information and the actual destination nodes for the stored information. The data pieces are then reassembled at the destination node. The nodes along the path between the source of the information and its destination are primarily responsible for making sure that each data piece received is transmitted on the correct outgoing link so that the data properly reaches its destination.

To properly meet user demands for information, a distributed network data storage system must provide high-availability of stored data to the computers needing the stored data (Pfister 1998). Specifically, a distributed network data storage system should be able to stay on-line with consistent availability of uncorrupted data, even if some hardware portion of the distributed network data storage system has crashed or becomes inaccessible because of an inability to transmit data. This is shown in Figure 1, where file pieces 3 and 5 have become inaccessible due to a hardware failure and a data transmission line break, respectively.

To address the requirement for high-availability of stored data, one or more variations of a data mirroring technique (U.S. Pat. No. 6,173,377, U.S. Pat. No. 6,157,991, U.S. Pat. No. 5,537,533) have been used in prior art data storage systems. In the execution of a data mirroring technique, crucial data is simply duplicated in its entirety at several locations in the distributed data storage system. Special care must be

taken to keep the data consistent across all locations where it is stored (U.S. Pat. No. 5,537,533). However, full mirroring of all data is costly both in hardware and physical time of transfer, particularly for large systems. One solution has been to keep the stored data consistent across all nodes, especially when the stored data could be changed on-line at several nodes simultaneously. This problem of keeping stored data consistent across all nodes in a data storage network is far from trivial.

There is little doubt that providing high-availability features in a distributed data storage system requires maintaining at least some level of redundancy of stored information. Historically, the problems associated with redundant data storage were addressed by the use of Redundant Arrays of Independent Disks (RAID) technology (Pfister 1998, Patterson *et al.*). The main concept behind RAID data storage technology is to divide the input data into units and then write/read several units of data simultaneously to several hard disk data storage systems. Several of the most commonly used configurations, or levels, of RAID arrays are described below.

The RAID Level 0 configuration implements a striped disk array for storing data. In a RAID Level 0 configuration, the data is broken down into blocks and each block is written to a separate data storage disk. The input/output performance of each disk drive is greatly improved by spreading the input/output load across many channels and disk drives. Reconstruction of the data set is accomplished by obtaining data blocks from each separate data storage disk.

The best data storage performance is achieved when the data to be stored is striped across multiple disk drives with each single disk drive attached to a single controller. No parity calculation overhead is involved, and there are no fault tolerance

capabilities in the RAID Level 0 configuration. There is no fault tolerance in the RAID Level 0 configuration because a single disk drive is connected to a single controller. Accordingly, the failure of just one disk drive will result in corruption of the stored data.

The RAID Level 1 configuration implements what is known as "disc mirroring." Disc mirroring is done to assure the reliability of stored data and a high degree of fault tolerance. A RAID Level 1 configuration also enhances data read performance, but the improved data read performance and fault tolerance come at the expense of available capacity in the disk drives used to store data. Specifically, the data to be stored is copied and then stored on multiple disk drives (or "mirrored"). The storage of data on multiple disk drives assures that, should one disk drive fail, the data is available from another disk drive on which the same data has been stored. The data read performance gain of a RAID Level 1 configuration can be realized if the redundant data is distributed evenly on all of the disk drives of a mirrored set within the subsystem. In a RAID Level 1 configuration, the number of data read requests and total wait state times both drop significantly. These drops are inversely proportional to the number of hard drives used in a RAID Level 1 configuration.

A RAID Level 5 configuration data storage algorithm represents a data storage methodology between a RAID Level 1 configuration and a RAID Level 0 configuration. The RAID Level 5 configuration is the last of the most common RAID data storage arrays in use, and is probably the most frequently implemented.

A RAID Level 5 configuration is really an adaptation of the RAID Level 0 configuration that sacrifices some data storage capacity for the same number of disk drives. However, the RAID Level 5 configuration gains a high level of data integrity or

fault tolerance. The RAID Level 5 configuration takes advantage of RAID Level 0's data striping methods, except that data is striped with parity across all of the disk drives in the array. The stripes of parity information are calculated using the "Exclusive OR" function. By using the Exclusive OR function with a series of data stripes in the RAID Level 5 configuration, any lost data can easily be recovered. Should any one disk drive in the array fail, the missing information can be determined in a manner similar to solving for a single variable in an equation (for example, solving for x in the equation, $4 + x = 7$). In an Exclusive OR operation, the equation would be similar to $1 - x = 1$. Thanks to the use of the Exclusive OR operation, there is always only one possible solution (in this case, 0), which provides a complete error recovery algorithm in a minimum amount of storage space.

A RAID Level 5 configuration achieves very high data transfer performance by reading data from or writing data to all of the disk drives simultaneously in parallel while retaining the means to reconstruct data if a given disk drive fails, thus maintaining data integrity for the data storage system.

A RAID Level 5 configuration minimizes the data write bottlenecks by distributing parity stripes over a series of hard drives. In doing so, a RAID Level 5 configuration provides relief to the concentration of data write activity on a single disk drive, in turn enhancing overall system performance.

The disadvantages of RAID-like implementation for distributed data storage systems are clear. First, it is impossible to dynamically control redundancy (classic RAID algorithms work in the case of failure of only one disk drive; if two or more disk drives go off line simultaneously, there is no way to recover data). Second, RAID

technology does not scale for more than ten disks, mainly due to the input/output intensive fault-recovery procedures which make the RAID technology unsuitable for systems where the unavailability of one or more nodes is common.

A similar data recovery problem arises when solving the problem of reliability of information transmission via communication channels. In this case algorithms of the Hamming error correction code (ECC) / error detection code (ECD) are usually used (Roman 1996). In general, there are two approaches to solving the problem of reliability of information transmission. Selecting a particular approach to solving this problem usually depends on requirements associated with the information transmission process. Both of the requirements associated with the information transmission process require transmitting redundant information to recover data in case of error. The first approach, called error-correction code (ECC), introduces redundancy into the stored information in the form of extra bits transmitted together with a data block so that it is possible to recover erroneous bits using received block and error-correction bits. The second approach, called error-detection code ECD, differs from the first approach in that one can only determine whether or not the data block contains errors without knowing which bits are incorrect.

One major drawback of both the error correction code and error detection code algorithms is that they are intended for data streaming recovery. Accordingly, these two algorithms carry a significant overhead in performance and amount of redundancy data. Even in case of errorless data transfer, one has to process a significantly larger amount of data than is necessary. Also, these two algorithms rely on the probability of a channel

error. In other words, these two algorithms work correctly only if the total number of errors in the received block of data does not exceed some predetermined number n.

Accordingly, there still remains a need in the art for a system which permits the storage of large amounts of data across a distributed arbitrarily-connected network of servers which provides high availability and fault tolerance.

SUMMARY

The present invention defines a system and method for the storage of data across a distributed arbitrarily-connected network of servers which provides high availability and fault tolerance. More particularly, the disclosed system and method enables the storage and retrieval of data with controllable redundancy. The controlled redundancy enables the optimal utilization of data storage facilities. Using the disclosed data storage system, it is possible to achieve an appropriate level of fault tolerance when either some of the servers in a network become inaccessible because of internal malfunctions or their connections to the data storage network are broken. According to the disclosed method for data storage, data file storing is allowed without total mirroring of the stored data on each server in the network.

Each complete set of data is broken into numbered, interchangeable data pieces of equal size, the number of which "n" may vary over time. However, from any "k" ($k \leq n$) number of data pieces, it is always possible to restore the data storage file completely. The use of this data storage system creates a condition of redundancy. The size of each data piece is about 1/k of the size of the entire stored file. The total number "n" of file pieces contained in a system may vary depending on the configuration of the arbitrary

network of servers and on the number of computers contained therein. However, the number of data pieces "n" is always equal to or greater than the number of data pieces needed to restore a file. While changing the number "n", the data stored in the existing pieces is not changed in any way. The number of data pieces may be large enough for modern computer networks and defines the selection of the storage algorithms.

The disclosed system and method for the storage of data across a distributed arbitrarily-connected network of servers also could be used for significant (up to "k" times) enlargement of the data transfer rate because of the fact that all pieces of the stored file could be transferred from "k" servers in parallel and independently from each other. Under some conditions, the disclosed data storage system and method provides the optimal utilization of network bandwidth and increases the rate of overall file transfer.

BRIEF DESCRIPTION OF THE DRAWING FIGURES

A better understanding of the data storage system and method of the present invention may be had by reference to the drawing figures wherein:

FIG. 1 is a schematic of a prior art system and method for information access on a distributed arbitrarily-connected network of servers showing the effect of a broken server or a break in a transmission line;

FIG. 2 is a schematic showing the system and method of the present invention for disassembling a file and storing file pieces on a distributed arbitrarily-connected network of servers;

FIG. 3 is a schematic showing the system and method of the present invention for file restoration; specifically, the collection of file pieces from the distributed arbitrarily-connected network of servers and their assemblage into the original file; and

FIG. 4 is a schematic showing the system and method of the present invention for additional generation of file pieces which could be done after initial distribution of data across the distributed arbitrarily-connected network of servers.

DETAILED DESCRIPTION OF THE INVENTION

The present invention defines a system and method for the storage of data files across a distributed arbitrarily-connected network of servers. Using the disclosed system and method, it is possible to achieve an appropriate level of data storage fault tolerance in a situation when some network servers become inaccessible because of internal malfunctions or breaks in the connection between servers in the network. The present invention provides a system and method which allows the storage of data files without a total mirroring of all data stored on each server in the network.

Each data file is proposed to be stored as a set of data pieces whose number "n" may vary over time. However, from any "k" number ($k \leq n$) of data pieces of the stored data file, it is always possible to restore the entire data file completely. The use of this data storage system creates a condition of redundancy. The size of each piece of the stored data file is about 1/k of the size of the entire stored data file. The number "n" of data pieces contained in the data storage system may vary depending on the configuration of the arbitrary network of servers and on the number of servers in the network. While changing the number "n", the data stored in the existing pieces is not changed in any way.

The first step in the disclosed method is to define an appropriate amount of data pieces P_1, P_2, P_3 ... P_n. A minimal amount "k" of data pieces needed to completely restore the file is defined usually from size requirements (for example, the size of one file piece could be taken close to optimal for transfer via Internet TCP/IP about 1 K byte).

The next step in the disclosed method is to define a fault tolerance level. For example, in a distributed arbitrarily-connected network including some number "L" working servers, fault tolerance is determined by setting the number "M" (M<L) of the working servers that could be switched off (or fail by themselves or due to network inaccessibility) simultaneously at any moment in time. To reconstruct the original file, at least M+k data pieces of the original file must be created and stored on at least M+k separate servers in the network. In such a case, the ability to restore the original file, even in case of the simultaneous unavailability of M servers, is guaranteed. This is because if M servers become unavailable, the required minimum number of data pieces to reconstruct the data set will still be available ((M+k) - M = k).

In Fig. 2 the process of file storing inside a distributed data storage system is shown. The first data file is split into at least M+k pieces as shown in Figure 2 by the blocks labeled P_1, P_2, P_3 ... P_n. In the second step, all of the created pieces are placed on different servers 1, 2, 3, 4 ... n in the distributed arbitrarily-connected network of servers **100**. Only after completion of such an operation will the step of storing the entire data file be treated as complete.

In Fig. 3 the process of retrieving the stored data from a distributed arbitrarily-connected data storage network **100** is shown. In the first stage of data retrieval, the required minimum number of data pieces "k" are received from the servers

1, 2, 3, 4...n on which the data pieces are stored. However, because not all servers are needed to obtain "k" data pieces, some of the servers, e.g., server 4 in Fig. 3, could be inaccessible because of internal failure or because of data transmission line problems. Because at least "k" servers are still available within network **100**, the entire data file can still be reconstructed. Therefore, in the first stage of data retrieval, as shown in the left side of Fig. 3, "k" data pieces would be collected, and in the second stage of data retrieval, as shown in the right side of Figure 3, the collected pieces are reconstructed into the original file on the client computer **200**.

In Fig. 4 the process of the additional generation of data pieces done after the initial distribution of data across a distributed arbitrarily-connected network **100** of servers 1, 2, 3, 4...n is shown.

During the normal operation of the disclosed network storage system, the requested amount of file pieces "n" could vary. For example, some servers could be permanently shut down or the requested level of fault tolerance increases. In such cases, additional data pieces must be created; that is, the number "n" must increase. To create an additional set of data pieces, as shown in the central portion of Fig. 4, an original file is assembled following the standard procedure for splitting the stored data file into pieces as shown in Fig. 2. Later, additional data pieces are generated **300** and distributed across a data storage network **400** on non-used servers. The amount of redundancy is increased by a quantum of 1/k of the original file size.

The disclosed method also could be used for a significant (up to "k" times) increase in data transfer rate because pieces of a data storage file could be transferred in parallel from "k" or more servers independently from each other. Under some conditions,

this method could give the optimal utilization of network bandwidth and speed up overall file transfer time.

While the disclosed invention has been described in terms of its preferred embodiment, those of ordinary skill in the art will understand that numerous other embodiments have been enabled. Such other embodiments shall be included within the scope and meaning of the appended claims.